PARIS - Knowledge Graph Alignment

Manuel Leone, Stefano Huber

Note about the experiment execution: performances evaluations of PARIS has been done using a Lenovo Thinkpad T450s with 12GB RAM and Intel I7-5600U CPU. This should be take in consideration especially when analyzing the total running time of PARIS. We experimented using more powerful machines which made the total running time of the experiment drop by more than half; in the end we decided to keep the measurements obtained with a simple laptop to underline the possibility to replicate such results using a common machine.

1 Introduction

The aim of this report is to analyze the performance of PARIS [3], a framework for Knowledge Graph (KG) alignment. The version used was the 0.3, which provides additional improvement to the JAR, being faster and more precise. Specifically, PARIS developers claim that their system is able to reach a precision of around 90% with some of the world largest ontologies.

Due to time and memory constraint we have not tested PARIS performances using such ontologies (the authors claim we need at least 24 GB of RAM to use it with such large datasets), so we will use a subset of DBpedia and Facebook composed by 15k entities each (DB15k and FB15k) [1]. These two datasets are extremely helpful as the author have already provided a `SameAs` dataset with the ground truth regarding the alignment of the two knowledge graphs.

The evaluation will be performed by randomly extracting a subset of the ground truth from the `SameAs` dataset to be used as seed for PARIS. The seeding is done by adding a *label* to the seed entities, following the format specified for the N-Triples format [2]. 3 different seeds percentage are used: *10%, 20%, 50%* and 20 runs are executed for each seed. For each of such run, four different metrics are computed: precision, recall, F_1 score and total running time.

We will now proceed to analyze the data we obtained from our experiments, followed by a conclusion on the analysis of PARIS.

2 Analysis of the results

2.1 Precision, Recall, F_1 score behaviour across the iterations and seeds

We firstly proceed to analyze how the precision, recall and F_1 score change across the iterations. Figure 2 shows how the different performance metrics change. What's immediately clear is that using an higher percentage of the ground truth clearly improves the performance (as expected), especially if considering the recall or the more general F_1 score. The precision has a slightly different trend only at the third iteration, where the 10% seed is surprisingly the best. Nevertheless, the precision shows a strange oscillating trend in the first iterations for all the seeds, starting from the perfect value of 1, dropping rapidly to 0.8-0.9 and then again increasing and decreasing one or two times before getting stable. Due to this, we can consider that having the 10% seed as top performer at the third iteration is just a consequence of this settlement phase, which is indeed confirmed by having again the 50% as the top one immediately after. We expect to have the 50% seed as top performer if compared with the 10% and 20% cases, since it is a bit like performing a machine learning algorithm using a super set of the training data: of course, the bigger the training data, the better the predictions.

The precision's oscillating trend at the first iterations is bound to the recall/F_1 score ones, which are almost a mirror, with few exceptions. First of all, at the iteration 0 the precision is the best we can have, **1** for all the seed cases. However, this high value must be considered together with the recall one, which is fairly low (0.1 for the 10% seed, 0.2 for the 20% and 0.5 for the 50%). The rationale is simple: PARIS add *only* the seed element in this iteration, and this explain both results. To better explain this fact it is better to look at how precision and recall are computed:

$$Precision = \frac{|\{ground\ truth\} \cap \{retrieved\ same\}|}{|\{retrieved\ same\}|}$$

$$Recall = \frac{|\{ground\ truth\} \cap \{retrieved\ same\}|}{|\{ground\ truth\}|}$$

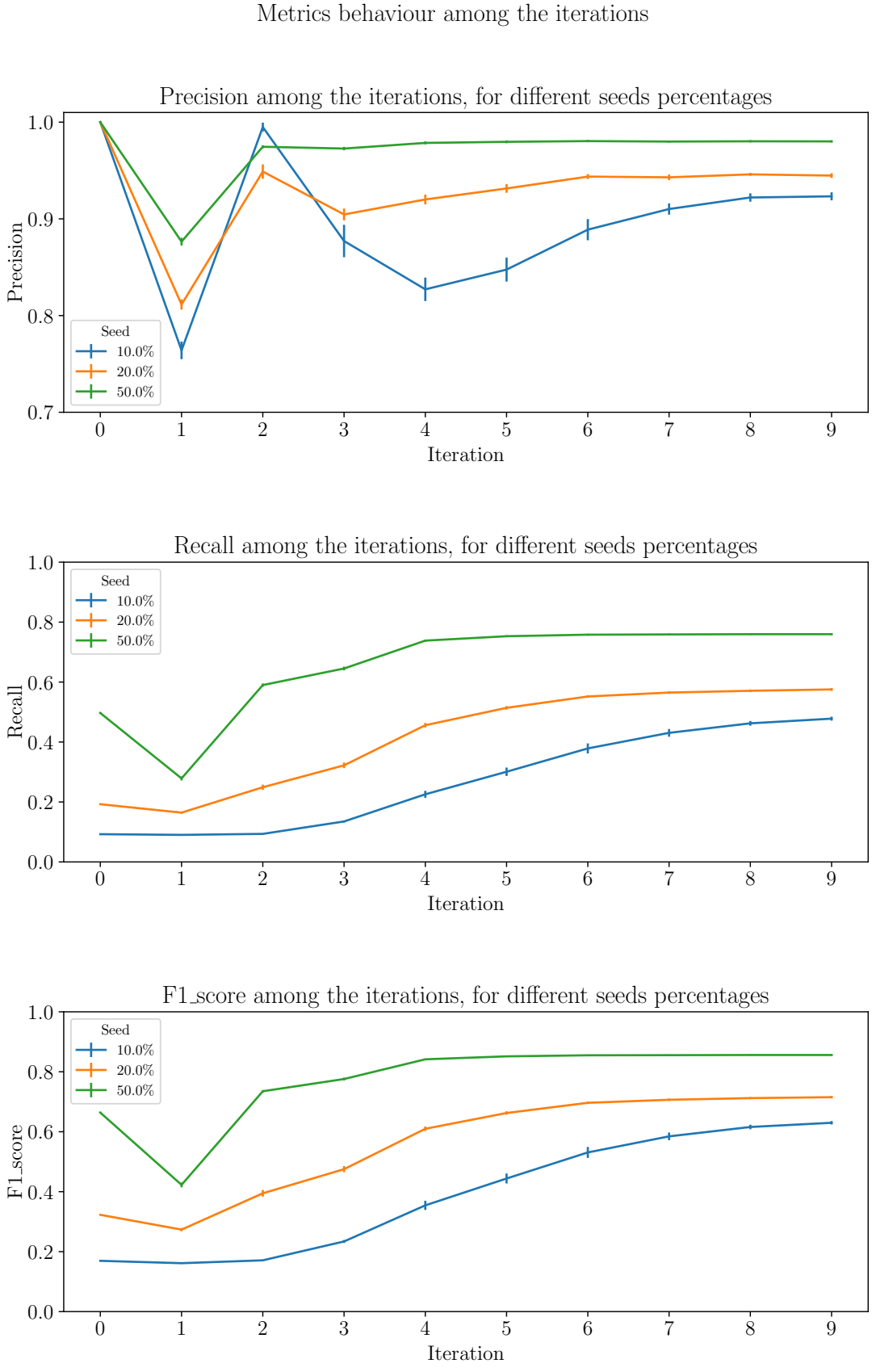

Figure 1: Precision, recall and F_1 scores behaviour against the iterations, with different seed

For the precision all the document retrieved are in the ground truth (as they are in the seed), so we divide the size of the retrieved by itself, achieving **1**. For the recall instead we get the same number as the numerator, but the denominator is the ground truth size, so we get exactly the same percentage we used as seed. Hence, the recall we obtain numerically is as much as the seed. For what concerns the F_1 score, this is of course low because it is pulled down by the low recall. The formula for the F_1 score is in fact

$$F_1 score = 2\frac{precision \cdot recall}{precision + recall}$$

and if we plot it with a fixed precision, it is clear that it gets highly influenced by low recalls (the F_1 gets small fast when the recall gets small).

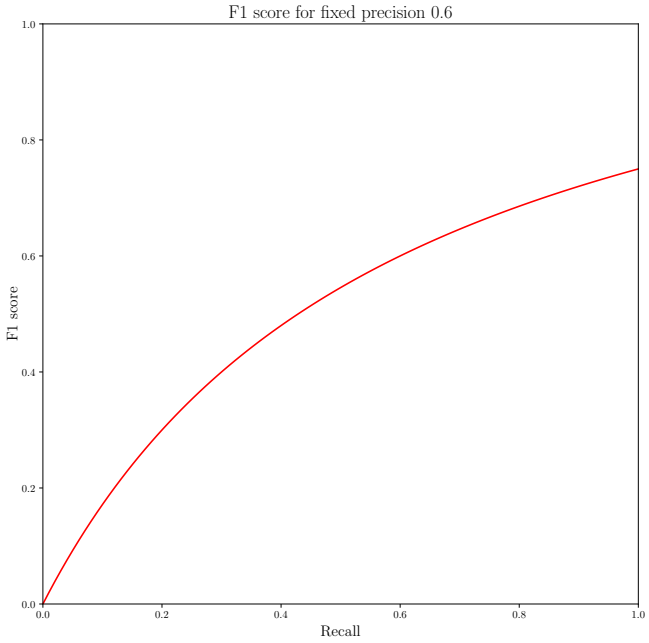

Figure 2: F_1 score trend for a fixed precision

If we now proceed further, considering the first 4/5 iterations, we can guess that PARIS now moves to retrieve more equalities between the two KGs. At the iteration 1 the algorithm guesses badly, causing both precision and recall to fall down (and F_1 score as a consequence). On the next iteration it improves both the metrics, then recall continues to increase, while the precision decreases. This is due to the fact that now the framework is increasing the size of the found equalities, but probably not at the same trend in which the total size of the retrieved ones increases. This is confirmed by the fact that lower is the seed percentage higher is the drop in precision. Overall, the algorithm is indeed improving, as confirmed by the F_1 score. Finally, from the fifth iteration on all the metrics improve, sign that now PARIS is more confident and it's guessing the equalities with higher probabilities.

Moreover, the plot error bars show the standard deviations at each iteration. We can see that the bars are all really small, with just some exceptions for the case in which we use 10% as seed. This confirms that even if PARIS is a probabilistic method it is really stable and reliable, providing a good reproducibility even if it runs with random subsets of the ground truth as seed.

To conclude, we want to point out a small remark: for all seeds, the last iterations is always the tenth. This indeed looks like a very strange coincidence, but we believe it is just a result of the inner functioning of PARIS itself, which is given to us a bit like a black box, since we easily execute it into its jar file. At the same time, since this method is not a proper machine learning technique, it should not suffer from overfitting problems, and hence even if it is trained for longer iterations then what are actually needed it should not affect the performance of the overall model (if it was, the 50% and 20% scores for all metrics' curves should start to diverge, instead of keeping constantly on the best F_1 score result). The same instead cannot be said for the 10% seed, which looks like it has not finished to converge at the tenth iteration. This again raises questions about how the number of iterations is chosen in the algorithm, which does not seem an ideal choice!

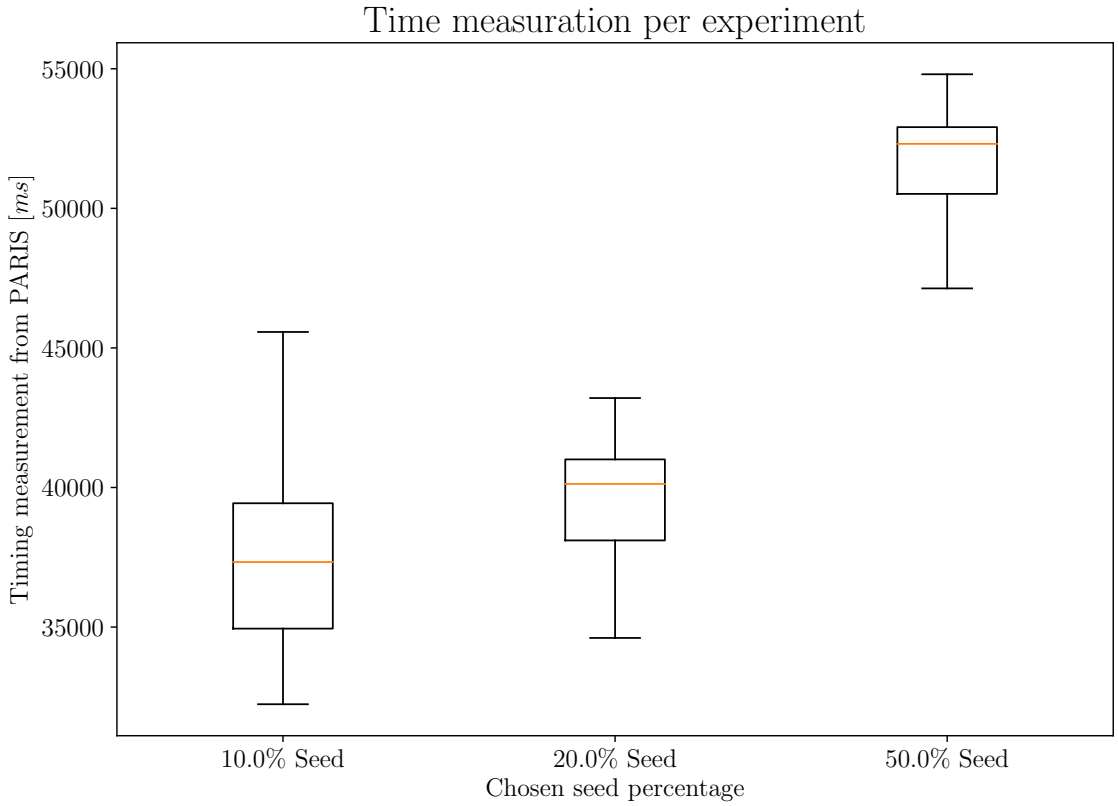

Figure 3: Total running time variation with different seeds

2.2 Running time for different seeds

Figure 3 shows how the total timing required by PARIS (in milliseconds) changes when considering different seed sizes. The boxplot enables us to see immediately two important things: the execution time increases when we use a bigger seed and there's a certain variability across the PARIS runs.

For the first point it's clear why PARIS runs in an higher time when considering a bigger seed: when the JAR is started PARIS runs a scan across both N-triples files to load all the object he can find for a given subject. Bigger is the file to scan, higher is the time necessary to completely do it. If the difference is subtle (like from using 10 or 20% of the truth), this increase in running time may be imperceptible (indeed, the two boxplots overlap with each other), while if the different is huge than the increase in timing may reach values of 15/20 seconds.

For what concerns the variability across the runs, the pattern is slightly different across the different seed sizes:

- The 10% case has a maximum which is really high with respect to all the other measurements, about 4 seconds away. This is not an outlier, but we can think that only few measurements are further from the third quantile, which is about 40 seconds. The median seems to be in the middle of Q1 and Q3, meaning that the measurements seem to be good distributed around it.
- The 20% and 50% cases have two similar trends (centered in different timings). Maximum and minimum are not so distant from the median and the box in general. Nevertheless, both seem to have a median that tends to be moved towards high values, meaning that when we increase the seed size PARIS tends to spend more time in general.

In order to better analyze such variability, mean and standard deviation for the timings across the runs for the different seeds are reported in Section 2.1. This proves what we stated before: the 10% seed has an higher variability as confirmed by its standard deviation, while the other two seed sizes have similar behaviour, with a low standard deviation.

Seed	μ **(s)**	σ **(s)**
10%	37.53	3.40
20%	39.6	2.11
50%	51.5	2.05

Table 1: Mean and standard deviation for the timings with different seed, computed on 20 runs per seed

To conclude, of course these timings need to be taken with a grain of salt, since we need to consider the fact that a laptop is not a perfect machine, and hence the CPU may suffer from hidden processes that the operating system executes automatically (so, even repeating the same experiment with an equally equipped machine, but running on a different OS, may cause the timings to vary deeply).

2.3 Metrics at last iteration

We now analyze how the three metrics (precision, recall and F_1 score) perform at the last iteration of the algorithm, for the different seed values. The last iteration, in fact, is indeed the most important one, since it is the iteration the algorithm PARIS has chosen to stop, since it has converged. To put it in a machine learning perspective, it is the best epoch for which we have trained the model, hence the performance at this epoch fully characterize the performance of the model we are going to use in production.

To analyze the three metrics, we first need to understand which values are better: of course, it is always difficult to judge which classifier is better based on precision or recall only. Assume for instance a classifier for edible mushrooms: in this case it is of extreme important that the precision is 1, or we would risk to kill somebody, no matter what is the recall (which, when high, would make us falsely discard some edible mushrooms, which is not too much a big loss). Similar reasoning can be made for the recall in analogous occasions. But, if we know both precision and recall, there is one case in which one classifier is clearly more powerful than the other, namely when both precision and recall are better for the same classifier. Usually it does not happen, since increasing the precision affects the recall, and vice versa, unless one classifier is actually much better than the other.

In this case, the classifiers are the same (PARIS), but simply trained on an increasing fraction of the training data: in case of the 10% seed, we try to predict the remaining 90% of sameAs relations based on the knowledge of only 10% of the data. When using the 50% seed, instead, we already know 50% of the ground truth, and try to predict the remaining 50%, which is clearly a much easier task! Hence, in this special case, we would expect

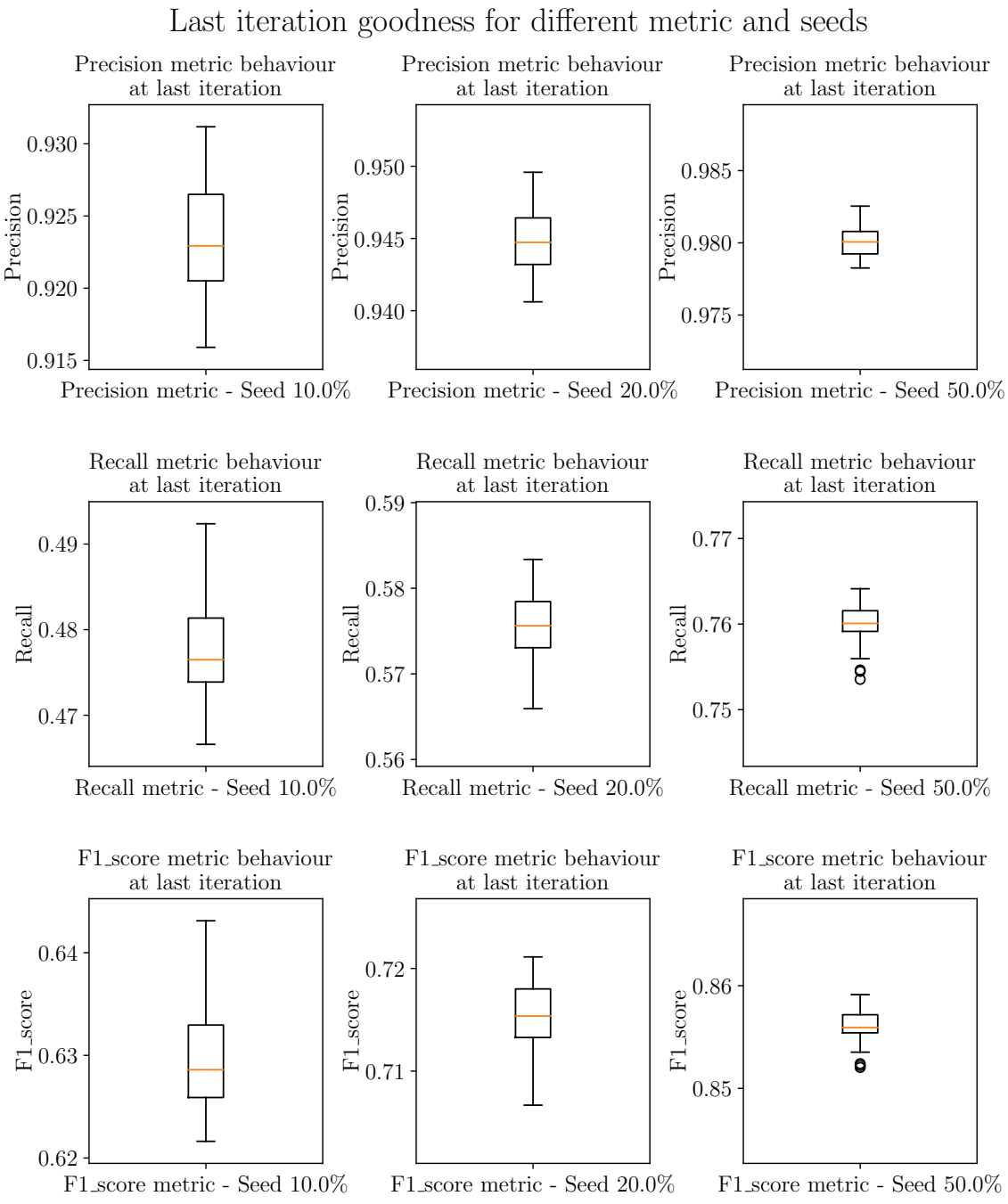
Last iteration goodness for different metric and seeds
Precision metric behaviour at last iteration
Precision
0.930
0.925
0.920
0.915
Precision metric - Seed 10.0%
Precision metric behaviour at last iteration
Precision
0.950
0.945
0.940
Precision metric - Seed 20.0%
Precision metric behaviour at last iteration
Precision
0.985
0.980
0.975
Precision metric - Seed 50.0%
Recall metric behaviour at last iteration
Recall
0.49
0.48
0.47
Recall metric - Seed 10.0%
Recall metric behaviour at last iteration
Recall
0.59
0.58
0.57
0.56
Recall metric - Seed 20.0%
Recall metric behaviour at last iteration
Recall
0.77
0.76
0.75
Recall metric - Seed 50.0%
F1_score metric behaviour at last iteration
F1_score
0.64
0.63
0.62
F1_score metric - Seed 10.0%
F1_score metric behaviour at last iteration
F1_score
0.72
0.71
F1_score metric - Seed 20.0%
F1_score metric behaviour at last iteration
F1_score
0.86
0.85
F1_score metric - Seed 50.0%

Figure 4

that the higher the seed, the more performing in both precision and recall the model is (and hence F_1 score, since it is a function with positive first derivative for both dimensions in the interval [0,1]).

Indeed, if we look at Figure 4, we notice that empirical evidence confirms our intuition: not only the precision and recall average is increasing for all three seeds, respectively $\approx 92\%$, $\approx 94.5\%$ and $\approx 98\%$ for precision and $\approx 48\%$, $\approx 57.5\%$ and $\approx 76\%$ for recall, but even the standard deviation gets smaller: although for every row of the plot in fig. 4 the y axes are differently centered, special attention has been paid to keep for all of them the same scale. Hence, the shrinker the boxplot is, the lower the standard deviation is, and hence the more consistent the measurements achieved have been. And indeed, if we notice carefully, on every row of precision and recall, the boxplots are increasingly getting smaller. So not only the precision and recall are higher the higher the seed is, but even more consistent. The boxplots for the F_1 score are a natural consequence of the previously computed two, in fact they respect the above trending (the higher the seed, the better and more consistent the measurements of the F_1 score is).

Seed	μ **(s)**	σ **(s)**
10%	0.92	0.0041
20%	0.94	0.0024
50%	0.98	0.0011

(a) Precision at last iteration

Seed	μ **(s)**	σ **(s)**
10%	0.47	0.0066
20%	0.57	0.0046
50%	0.75	0.0029

(b) Recall at last iteration

Seed	μ **(s)**	σ **(s)**
10%	0.62	0.0057
20%	0.71	0.0038
50%	0.85	0.0020

(c) F1 score at last iteration

2.4 Confidence interval for the metrics

These plots have been created bootstrapping the observations: we have computed 20 different experiments for every seed, hence we have 20 different values for the precision, recall and F_1 score. In order to judge whether those observations are consistent around the mean, we can perform bootstrapping: we sample allowing repetitions the 20 observations in groups of 20, for 1000 times. Then we compute the mean of every bootstrapped sample, and collect such observations in a histogram. We can see that, as expected, the result for the bootstrapped mean of every metric is a gaussian. Moreover, we have plotted the 95% confidence interval, which should be as close as possible when the measurements are very consistent. Once again, we decided to keep the same x axis scale for the same row, and hence we can see graphically that indeed the confidence intervals are closer, when the seed is higher, which means that the confidence intervals are tighter, and the observations more consistent.

This result was expected, since in section 2.3 we have noticed that the boxplots get shorter, and hence the measurements more consistent, with higher seeds.

2.5 Confidence interval for the timings

To conclude, we analyze fig. 6, the confidence interval for the timings. As in the case of section 2.4, we have obtained such plots by bootstrapping the timing observed during the 20 experiments we have done.

As already mention and explained in section 2.2 we can observe that the higher the seed, the longer it takes. But, this time, it looks like the higher the seed, the more consistent around the mean the results are. This result is quite surprising in fact we would have expect that the longer the execution time is, the more frequent the Operating System interference described in section 2.2 are. Hence, we would have expected that the standard deviation would have kept unaltered or even got bigger, but empirically it looks like it is not the case.

3 Conclusions

We have empirically tested the performance of the algorithm PARIS on the task of performing knowledge graphs alignment. The datasets we have used are a small subset (15k rows) of the two more complex DBpedia and Facebook knowledge graphs.

To have a comparison, we have taken advantage of human-labeled ground truth in terms of alignment relations and using a small subset of such ground truth as seed (10%, 20% and 50%), we used PARIS in order to predict the remaining sameAs relations. In order to have more solid results, we have repeated each experiment for every seed multiple times (20), in order to have a better estimator (with both the mean and the standard deviation) for the mean performance in terms of precision, recall, F_1 score and time of execution.

As expected, the higher the seed, the higher the precision, the recall and the F_1 score are, and even the more consistent the measurements (with low standard deviation): this result is in line with our expectations, since it is easier for the algorithm to guess missing sameAs relations when it knows already a lot of them, rather then trying to guess based on a very little information basis.

Computed confidence intervals at the last iteration, for different seeds/metrics

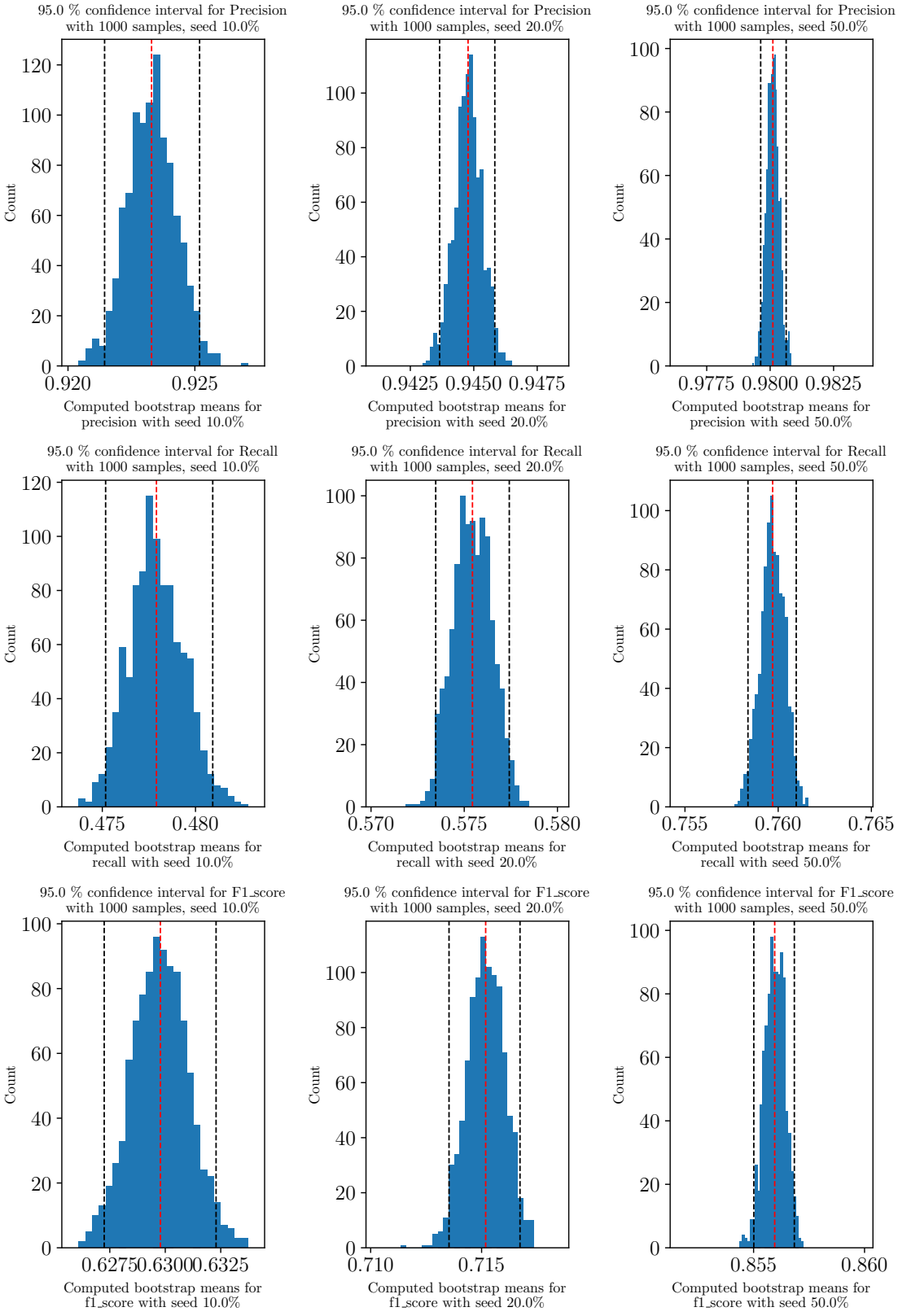

Figure 5

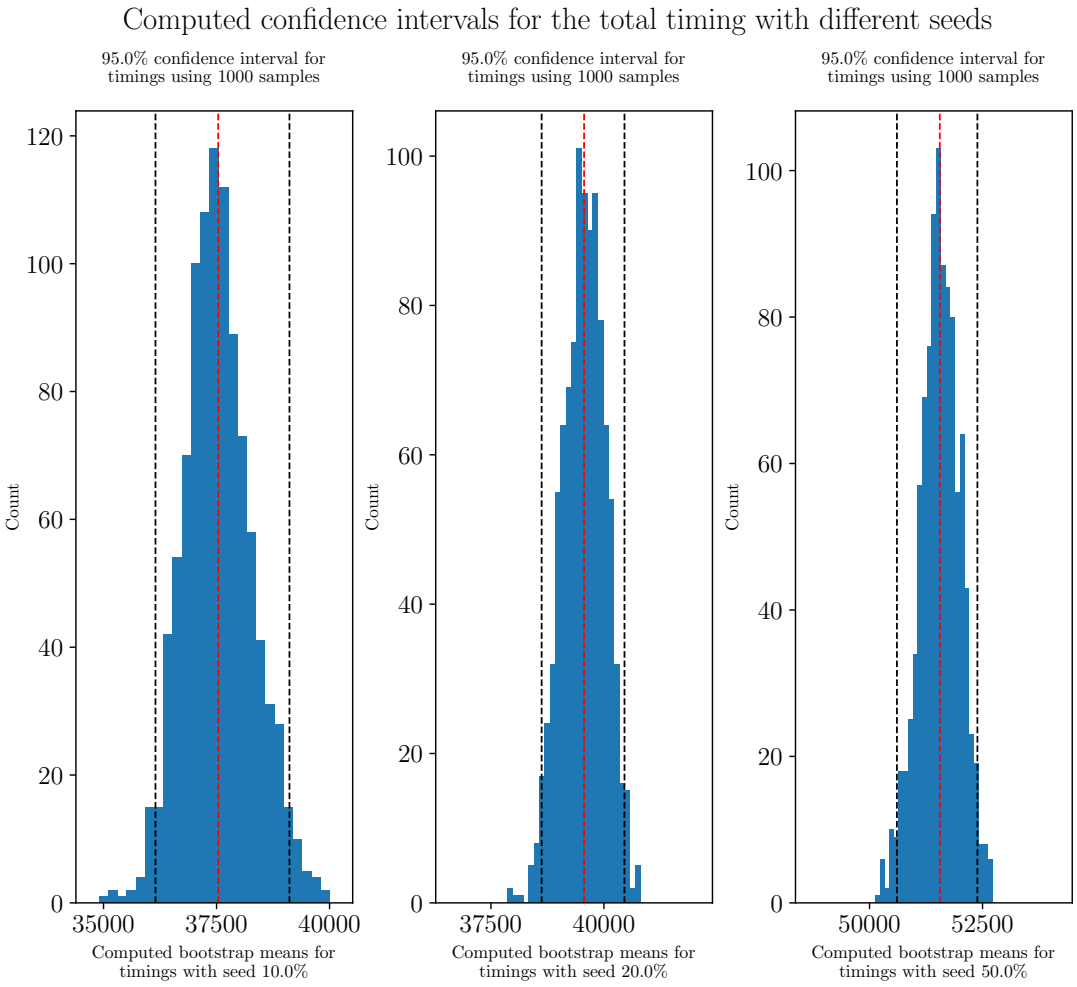
Computed confidence intervals for the total timing with different seeds
95.0% confidence interval for timings using 1000 samples
95.0% confidence interval for timings using 1000 samples
95.0% confidence interval for timings using 1000 samples
Count
Count
Count
0
20
40
60
80
100
120
35000
37500
40000
37500
40000
50000
52500
Computed bootstrap means for timings with seed 10.0%
Computed bootstrap means for timings with seed 20.0%
Computed bootstrap means for timings with seed 50.0%

Figure 6

Even the time required to execute the algorithm reflects what are our initial hypothesis: the higher the seed, the more data the algorithm needs to process, and hence the higher the time required is. But, surprisingly, the higher the seed, and the more consistent the results are around the average time: this result was unexpected, as briefly explained in section 2.5, and may need further analysis to model how the noise in the timing measurements is estimated.

References

[1] NLE-ML. *Multimodal Knowledge Graphs*. URL: https://github.com/nle-ml/mmkb. (accessed: 18.06.2020).

[2] W3 Organization. *RDF 1.1 N-Triples - W3C Recommendation 25 February 2014*. URL: https://www.w3.org/TR/n-triples/. (accessed: 18.06.2020).

[3] Webdam team. *PARIS - Probabilistic Alignment of Relations, Instances, and Schema*. URL: http://webdam.inria.fr/paris/. (accessed: 18.06.2020).